# CodeWarrior®

# ZoneRanger User Guide

Because of last-minute changes to CodeWarrior, some of the information in this manual may be inaccurate. Please read the Release Notes on the CodeWarrior CD for the latest up-to-date information.

Revised: 98/08/24 raw

Metrowerks CodeWarrior copyright ©1993–1998 by Metrowerks Inc. and its licensors. All rights reserved.

Documentation stored on the compact disk(s) may be printed by licensee for personal use. Except for the foregoing, no part of this documentation may be reproduced or transmitted in any form by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from Metrowerks Inc.

Metrowerks, the Metrowerks logo, CodeWarrior, and Software at Work are registered trademarks of Metrowerks Inc. PowerPlant and PowerPlant Constructor are trademarks of Metrowerks Inc.

All other trademarks and registered trademarks are the property of their respective owners.

ALL SOFTWARE AND DOCUMENTATION ON THE COMPACT DISK(S) ARE SUBJECT TO THE LICENSE AGREEMENT IN THE CD BOOKLET.

## How to Contact Metrowerks:

| | |
|---|---|
| **U.S.A. and international** | Metrowerks Corporation<br>9801 Metric, Suite #100<br>Austin, TX 78758<br>U.S.A. |
| **Canada** | Metrowerks Inc.<br>1500 du College, Suite 300<br>Ville St-Laurent, QC<br>Canada H4L 5G6 |
| **Ordering** | Voice: (800) 377–5416<br>Fax: (512) 873–4901 |
| **World Wide Web** | `http://www.metrowerks.com` |
| **Registration information** | `register@metrowerks.com` |
| **Technical support** | `support@metrowerks.com` |
| **Sales, marketing, & licensing** | `sales@metrowerks.com` |
| **CompuServe** | goto `Metrowerks` |

# Table of Contents


**1 Introduction** **5**

Read the Release Notes . . . . . 5
Metrowerks Year 2000 Compliance . . . . . 6
What's New in This Release . . . . . 6
CodeWarrior and Its Documentation . . . . . 6
What's in This Manual . . . . . 7
Where to Go from Here . . . . . 8

**2 Getting Started** **9**

System Requirements . . . . . 9
Installing ZoneRanger . . . . . 10
Macintosh Memory Architecture . . . . . 10
What Is a Heap Zone? . . . . . 11
What Does a Process Do With a Heap Zone? . . . . . 11
What's Inside a Heap Zone? . . . . . 11
About Pointers and Handles . . . . . 12

**3 What You See** **15**

What You See Overview . . . . . 15
Heap Hierarchy Window . . . . . 16
Summary Display . . . . . 17
Graphical Windows . . . . . 18
Identifying Parts of the Graphical Windows . . . . . 19
1D Graphical Display . . . . . 20
2D Graphical Display . . . . . 22
Histogram Display . . . . . 24
Block List Window . . . . . 26
Block Window . . . . . 28

**4 Using ZoneRanger** **31**

Using ZoneRanger Overview . . . . . 31
Viewing a Process's Heap . . . . . 32
Opening a Graphical or Summary Display . . . . . 32
Viewing Block information in Graphical windows . . . . . 33

Viewing Information on a Heap Zone and Block . . . . . 36
Heap Navigation . . . . . 38
Searching for Blocks. . . . . 40
Special Operations . . . . . 43
Choosing How Often to Refresh Contents . . . . . 43
Compacting and Purging Heaps . . . . . 44
Going To and Returning From a Debugger. . . . . 45
Finding Memory Leaks . . . . . 45
Finding Memory Requirements . . . . . 48
Browsing `malloc` Heaps . . . . . 50
Quitting and Relaunching an Application . . . . . 50

**5 ZoneRanger Menus 51**

ZoneRanger Menus Overview . . . . . 51
Apple menu . . . . . 51
File menu . . . . . 52
Edit menu . . . . . 53
Configure menu . . . . . 55
Special menu. . . . . 58
Window menu . . . . . 60

**Index 61**

# 1 Introduction

Welcome to the CodeWarrior ZoneRanger manual. ZoneRanger is a dynamic memory inspection tool that allows you to study how any application—especially one you are creating—uses or misuses memory. This manual discusses how to install and use the MW ZoneRanger application.

The topics in this chapter are:

- Read the Release Notes—where to go for critical, last-second details
- Metrowerks Year 2000 Compliance—what you need to know about this software's Year 2000 compliance
- What's New in This Release—new features since the previous release
- CodeWarrior and Its Documentation—a general description of the CodeWarrior architecture and documentation
- What's in This Manual—a description of the contents of this manual
- Where to Go from Here—an overview of the chapters in this manual

## Read the Release Notes

Before you use ZoneRanger, you should read the release notes. They contain important last-minute information about new features, bug fixes, and incompatibilities that *may not be included in the documentation*.

The release notes directory is always included as part of a standard CodeWarrior installation. The release notes directory is also located at the top level of the CodeWarrior CD.

# Metrowerks Year 2000 Compliance

The Products provided by Metrowerks under the License agreement process dates only to the extent that the Products use date data provided by the host or target operating system for date representations used in internal processes, such as file modifications. Any Year 2000 Compliance issues resulting from the operation of the Products are therefore necessarily subject to the Year 2000 Compliance of the relevant host or target operating system. Metrowerks directs you to the relevant statements of Microsoft Corporation, Sun Microsystems, Inc., Apple Computer, Inc., and other host or target operating systems relating to the Year 2000 Compliance of their operating systems. Except as expressly described above, the Products, in themselves, do not process date data and therefore do not implicate Year 2000 Compliance issues.

For additional information, visit: `http://www.metrowerks.com/about/y2k.html`.

# What's New in This Release

ZoneRanger now includes the ability to browse "embedded" heap zones created via `InitZone`. This feature requires MacOS for PowerPC.

The old "Heap List" window has been replaced by the new "Heap Hierarchy" window. This tree-structured, hierarchical listing uses the familiar turn-down-triangle metaphor used by the Finder to associate embedded heaps to their parent heaps

# CodeWarrior and Its Documentation

CodeWarrior is a multi-host, multi-language, multi-target development environment. What does that mean?

**Multiple hosts** CodeWarrior runs on several different operating systems, including Windows and Mac OS. The features, human interface, and operation of CodeWarrior are very similar on all hosts.

**Multiple languages** You can use CodeWarrior to program in several languages, including C/C++, Pascal, and Java. Third-party compilers provide support for other languages, such as Fortran. Which languages are available to you depend upon the target for which you are developing software.

**Multiple targets** You can use CodeWarrior to write software for several different chips or operating systems. CodeWarrior products support programming for a variety of embedded processors and real-time operating systems, the Java Virtual Machine, and desktop operating systems such as Windows, Mac OS, and BeOS.

Most features of CodeWarrior apply regardless of your preferred host, language, or target. General features of CodeWarrior are described in manuals such as the *IDE User Guide* and *Debugger User Guide*.

For a complete understanding of CodeWarrior, you must refer to both the general documentation and the documentation that is specific to your particular target.

# What's in This Manual

Table 1.1 lists every chapter in this manual, and describes the information contained in each. See "CodeWarrior and Its Documentation" on page 6 for a discussion of how these chapters relate to other CodeWarrior documentation.

**Table 1.1 Contents of chapters**

| Chapter | Description |
|---|---|
| Introduction | this chapter |
| Getting Started | system requirements, installation instructions, and background information |
| What You See | all about ZoneRanger's interface |

| Chapter | Description |
|---|---|
| Using ZoneRanger | how to use ZoneRanger's functions |
| ZoneRanger Menus | a description of each menu command available in ZoneRanger |

# Where to Go from Here

If you are new to Mac OS programming and are unfamiliar with the underlying Mac OS memory architecture and the concepts of relocatable and non-relocatable blocks, you should start reading the information in "Getting Started" on page 9. This section introduces you to terms and explains the basic concepts used in ZoneRanger.

If you are unfamiliar with ZoneRanger, read "What You See" on page 15, and "Using ZoneRanger" on page 31. These sections introduce you to the various windows, displays, and options you have available in ZoneRanger, and how to use them to accomplish specific tasks.

"ZoneRanger Menus" on page 51 is an excellent starting point for new and experienced users alike. This section introduces you to each of ZoneRanger's menu commands in detail.

If you are already familiar with ZoneRanger, use this manual as reference material when you have specific questions. There is also a complete index included in this manual.

**See also** *Inside Macintosh: Memory* and *Inside Macintosh: PowerPC System Software*.

# 2

# Getting Started

This chapter gives you the information you need to get ZoneRanger installed and running. For new users, this chapter discusses some basic principles of memory management in the Mac OS environment, including heap zones, relocatable blocks, non-relocatable blocks, handles, pointers, and memory allocation.

This chapter includes the following topics:

- System Requirements — hardware and software requirements
- Installing ZoneRanger— how to install ZoneRanger on your system
- Macintosh Memory Architecture — an introduction to Mac OS memory management

## System Requirements

If you can run CodeWarrior, you can run ZoneRanger. ZoneRanger requires a Motorola 68020 processor or better, or a PowerPC 601 or better processor. ZoneRanger needs approximately 1 MB of disk space and 640 KB of free RAM.

ZoneRanger requires System 7.1 or later (for 68K Mac OS systems) or System 7.1.2 or later (for PowerPC Mac OS systems), and Color QuickDraw.

ZoneRanger is a stand-alone application meant to help you analyze how your program is managing memory. ZoneRanger is not dependent on any part of the CodeWarrior IDE.

# Installing ZoneRanger

Installing ZoneRanger is simple. Choose one of two methods: automatic or manual.

The CodeWarrior Installer automatically installs ZoneRanger by choosing any of the following installations:

- CodeWarrior Heaven—absolutely everything for the CodeWarrior IDE
- Mac OS Heaven—everything you need to use CodeWarrior for the Mac OS
- Other Metrowerks Tools—the complete selection
- ZoneRanger—from the Other Metrowerks Tools options

The Discover series has slightly different install options. You can install ZoneRanger by choosing any of the following installations:

- Easy Install—absolutely everything for the CodeWarrior IDE
- ZoneRanger—from the Other Metrowerks Tools option in the Custom Install panel

To install ZoneRanger manually, locate the ZoneRanger folder on the installation CD. It is in the Other Metrowerks Tools folder. Copy the folder to your hard drive. There are no other files required.

# Macintosh Memory Architecture

A process is a running task of any type. A running application is a process. So is a desk accessory, the System, and other services that run in the background. Each process in the Mac OS has an associated heap zone.

The rest of this section discusses the following topics:

- What Is a Heap Zone?
- What Does a Process Do With a Heap Zone?
- What's Inside a Heap Zone?
- About Pointers and Handles

## What Is a Heap Zone?

A heap zone is a range of contiguous addresses in the computer's memory dedicated to a process. In other words, it's a single chunk of RAM assigned to a process. The Process Manager creates the heap for a process when it launches the process. The size of the heap is based on the process's requested memory requirements (which you can see in an application's Get Info box in the Finder). If you are writing your own application, you can set the memory requirements in the PPC Project or 68K Project panel in the Project Settings dialog box.

---

**NOTE:** Not all processes have to be applications. Some extensions also have a process associated with them. Extensions that display background screens are a good example. There is no way of changing the requested memory requirements of Extension processes. There are utilities that will let you see the memory requirements, however.

---

## What Does a Process Do With a Heap Zone?

The heap zone is where a process works. For example, as an application does its work, it must create and dispose of blocks of memory for its own purposes. The application uses these blocks to store data. The Memory Manager allocates memory blocks in the heap zone on behalf of the process.

As with all rules there are exceptions, but in general everything a typical process does with memory happens inside its own heap zone.

## What's Inside a Heap Zone?

There is no empty space in a heap zone. A heap zone consists of a zone header followed by a series of memory blocks. In turn, each individual block has a block header followed by the contents of the block. A block header contains the size, type, and attributes of the block.

The block size is the number of bytes in the block. A block has a physical size and a logical size. The physical size is the block header plus the size of the contents. There may also be a few unused "pad" bytes at the end of the block, because the Mac OS Memory Manager allocates space in even numbers of bytes. For example, if the block of memory you allocate is 15, the Memory Manager rounds the size to 16. The minimum number of bytes in a block is 12. The logical size is the size of the contents only.

There are three types of blocks. A block may be relocatable, non-relocatable, or free.

- **relocatable block** —one that the Memory Manager can move around the heap zone as needed. It contains data that is referenced by a **handle** .
- **non-relocatable block** —one that cannot move around the heap zone. It contains the data that is referenced by a **pointer** .
- **free block** —available to the Memory Manager for reallocation into relocatable or non-relocatable blocks. The contents of a free block are undefined.

Relocatable blocks have attributes. There are three block attributes: locked, purgeable, and resource.

- **locked handle** —cannot be moved around the heap zone.
- **purgeable handle** —can be converted to a free block if necessary.
- **resource handle** —contains the data from a resource.

## About Pointers and Handles

A non-relocatable block never moves, so the address never changes. When you allocate a non-relocatable block, the Memory Manager gives you the address of the block—a pointer to the block.

When you allocate a relocatable block, the Memory Manager stores the address of the data in a special location called a master pointer. Every time the relocatable block moves, the Memory Manager updates the contents of the master pointer, so the master pointer always points to the right address for the data. When you allocate the

relocatable block, the Memory Manager returns *the address of the master pointer* to you. This pointer to the master pointer is a handle. To find out where the data is, you look in the handle to get the master pointer, then look in the master pointer to find out where the data is at that moment.

The purpose behind relocatable blocks is simple. Because an application's memory demands vary from one moment to the next, it makes sense to be able to shuffle blocks of memory to make the most efficient use of available space. There are many times when the ability to move memory around prevents a program from crashing or being unable to function. If the Memory Manager didn't shuffle blocks of memory for you, you would have to do it yourself.

**See also** *Inside Macintosh: Memory* for more information on the structure of heap zones and memory blocks.

# 3

# What You See

This chapter discusses the various windows you see when using ZoneRanger. More detail on how to use ZoneRanger can be found in "Using ZoneRanger" on page 31.

## What You See Overview

Each process usually has its own memory space called a heap zone. ZoneRanger displays detailed information about any heap zone currently active on a Mac OS system. It displays the number and sizes of free blocks, pointers, handles, locked handles, purgeable handles, and resource handles. It can also graphically display a heap zone, with the size, type, and attributes of each block clearly marked.

ZoneRanger has five types of windows:

- Heap Hierarchy Window—displays a structured list of currently open heap zones
- Summary Display—a summary of the number and total size of each type of block in a given heap zone
- Graphical Windows—various graphical representations of a given heap zone
- Block List Window—detailed information about a specific set of blocks in a given heap zone
- Block Window—the entire contents of a given block in a given heap zone

ZoneRanger updates all windows at an interval you choose. The only exception is the Block List window, whose contents are static.

**TIP:** ZoneRanger provides complete balloon help for all menu items, windows, and dialogs. You can view menu item balloons

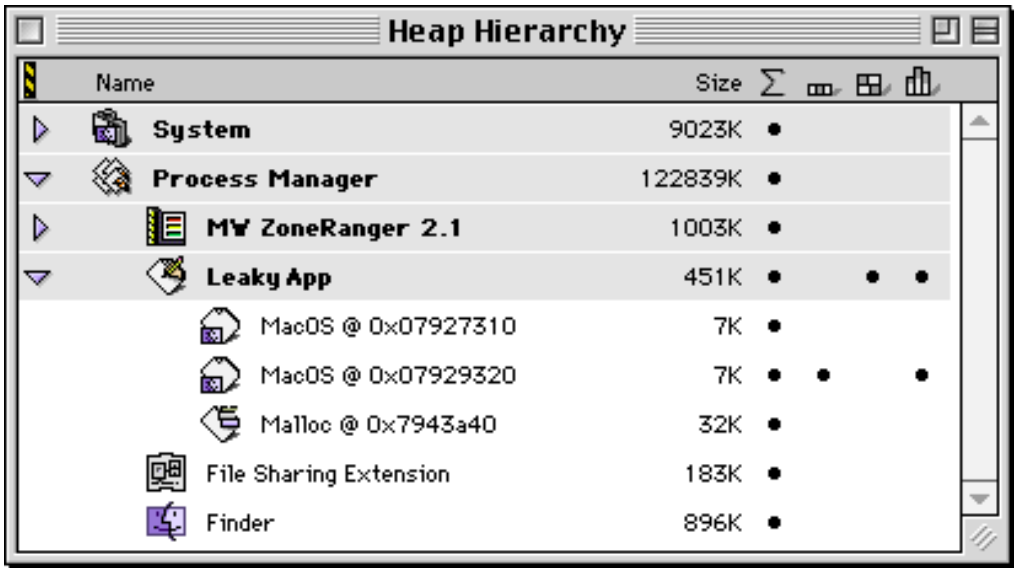
without turning on balloon help. Hold down the Option key as you click in a menu. ZoneRanger displays balloon help until you release the mouse button.

If ZoneRanger runs low on memory, commands that require additional memory are disabled. To make more memory available, close any unneeded windows.

# Heap Hierarchy Window

When launched, ZoneRanger automatically displays the Heap Hierarchy window shown in Figure 3.1. This window lists all available processes and the heap zones associated with them. You may display this window at any time by choosing **Open** from the File menu.

**Figure 3.1 The Heap Hierarchy window**



You do not need to close the Heap Hierarchy window after opening a heap zone. Left open, the window will reflect the changes in the heap zones as applications are launched and closed.

The Heap Hierarchy has four columns of bullets that control the presence of the different heap display windows. These bullets are described in Table 3.1.

**Table 3.1 Heap Hierarchy bullet columns**

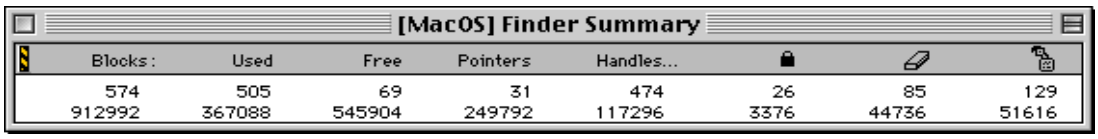
| This Bullet… | Opens this display… |
|---|---|
| Σ | Summary display |
| | One-dimensional graphical display |
| | Two-dimensional graphical display |
| | Histogram display |

**See also** "Choosing How Often to Refresh Contents" on page 43, "Quitting and Relaunching an Application" on page 50 and "Limitations" on page 47.

# Summary Display

The Summary display shows information on all the blocks in an application's heap.

**Figure 3.2 The Summary window**



The Summary Display has two rows of numbers. The first row contains the total number of the blocks of each type. The second row contains the total size of the blocks in bytes. The column headings are described in Table 3.2.

**Table 3.2 Summary window columns**

| This Column… | Contains the number or total size of… |
|---|---|
| Blocks | All blocks |
| Used | Used blocks (where "used" means not free) |
| Free | Free blocks |
| Pointers | Pointers |
| Handles | Handles |
| | Locked handles |
| | Purgeable handles |
| | Resource handles |

To see detailed information about the blocks in a column, double click on that column, ZoneRanger opens a Block List Window.

You can copy all of the information in the Summary Display to the clipboard, or save the same information to a text file.

**See also** "Block List Window" on page 26.

# Graphical Windows

Each graphical window displays all the blocks in an application's heap. The name of the window corresponds to the name of the pro-

cess it is displaying. The function of each of the graphical windows are identical.

This section discusses the following topics:

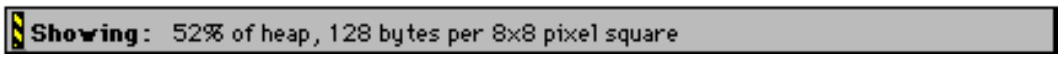
- Identifying Parts of the Graphical Windows—describes various parts of the graphical windows
- 1D Graphical Display—displays a process's heap horizontally (one dimensional)
- 2D Graphical Display—displays a process's heap horizontally and vertically (two dimensional)
- Histogram Display—shows a process's heap as a histogram

## Identifying Parts of the Graphical Windows

There is a lot of information displayed in each graphical window. It is important to understand the various parts of the graphical windows in order to use them effectively.

The grey area at the top of the window (Figure 3.3) tells you what percentage of the application's heap you see in the window. This area also tells you resolution and zoom factor you are currently viewing at.

**Figure 3.3 Heap information strip**




If you select a block in any Graphical window, summary information about that block is shown at the bottom of the window beside the popup menus (Figure 3.4). This information tells you the type of block, its address, size in bytes, and its attributes (if the selected block is a handle). This information can also be found in the Block Information popup menu.

**Figure 3.4 Block information strip**

In many ZoneRanger windows, not just the graphical windows, there is a series of up to three icons in the lower left corner of the window. These icons are popup menus that display information about the current zone, selected block (if any), and color or pattern legend. Table 3.3 lists the icons the windows in which they appear.

**Table 3.3 Information menus**

| Icon (scaled) | Description | Available in... |
|---|---|---|
| | Zone Information | 1D Graphical Display, 2D Graphical Display, Histogram Display, Block List Window, Block Window, and the Find dialog box |
| | Block Information | 1D Graphical Display, 2D Graphical Display, and Block Window |
| | Legend Information | 1D Graphical Display, 2D Graphical Display, and Histogram Display |

The rest of the window is a graphical representation of the heap zone for the application under inspection.

**See also** "Searching for Blocks" on page 40, "Block Information popup menu" on page 37, "Zone Information popup menu" on page 36, and "Changing the Graphical windows' colors" on page 34.

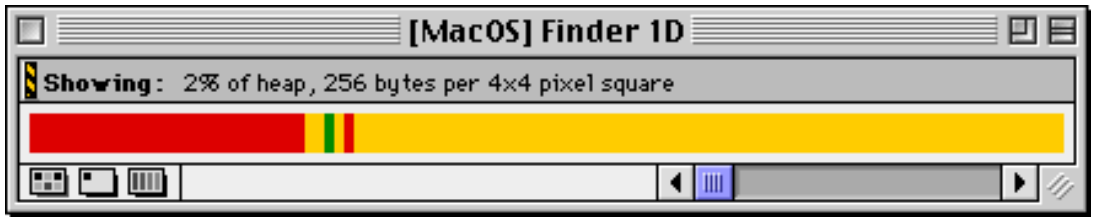
**TIP:** Switch between the Graphical and Summary views for a heap zone with Command-Tab.

## 1D Graphical Display

The 1D Graphical Display shows a process's heap zone as contiguous sections that look similar to a stacked bar graph (Figure 3.5). Each block represents a fixed number of bytes in the heap zone. This

number is called the resolution. If a block is smaller than the resolution, it occupies less than a square, and may not appear in the window at all. To change the resolution, use the **Increase Resolution** and **Decrease Resolution** commands in the Configure menu. If you want to force the resolution to the size of the window, use the **Fit To Window** command. You cannot set the resolution to less than 8 bytes or more than 1 megabyte per square. The default resolution is 256 bytes.

**Figure 3.5 The 1D Graphical display**




Each block takes up a certain number of pixels in the window. To change the size, use the **Zoom In** and **Zoom Out** commands in the Configure menu. By default, the each square is 1 pixel by 1 pixel. If the block size is greater than 1 pixel by 1 pixel, you can see the block outlines by using the **Show/Hide Block Outlines** commands in the Configure menu. Outlining helps you get an overall view of how many blocks are in each heap zone and how large they are.

Each type of block is drawn with a different pattern (if the window is in monochrome) or color (if the window is in color). By default, ZoneRanger uses color on color monitors and monochrome on monochrome monitors. To change this setting, choose **Use Color** or **Use Monochrome** from the Configure menu.

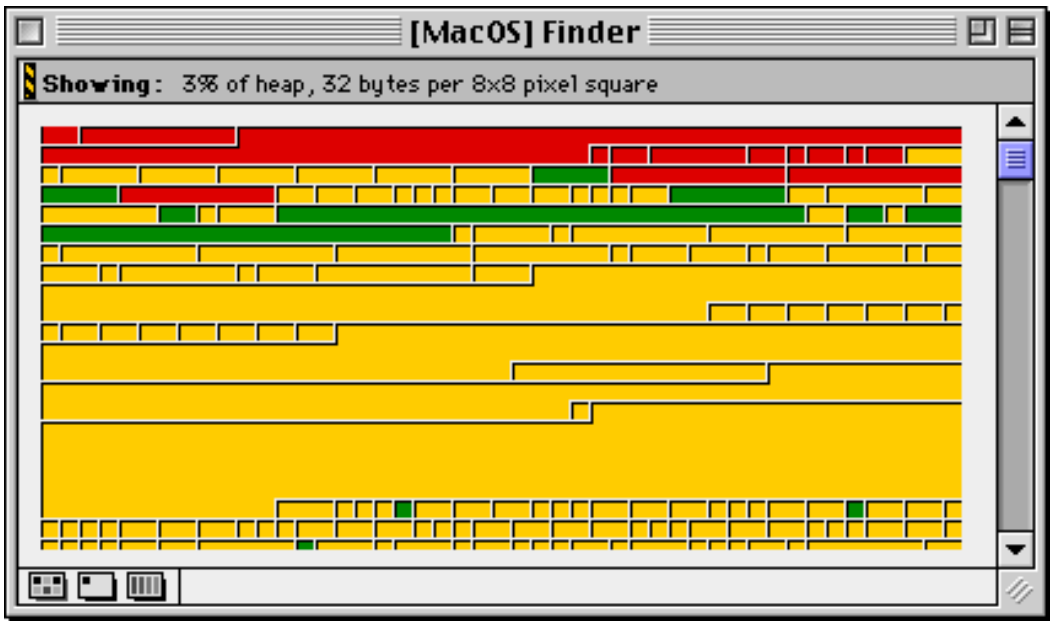
In the default color display, free blocks are green, handles are yellow, and pointers are red. If the display is monochrome, free blocks are white, handles are gray, and pointers are black. To change the colors for the blocks, use the Legend Information popup menu. You cannot change the patterns used in a monochrome display.

**See also** "Changing the Graphical windows' colors" on page 34.

## 2D Graphical Display

The 2D Graphical Display shows a process's heap zone, wrapped to the width of the window. The display is actually a series of small squares (Figure 3.6).

**Figure 3.6 The 2D Graphical display**




Each square represents a fixed number of bytes in the heap zone. This number is called the resolution. If a block is larger than the resolution, it spans several squares. If a block is smaller than the resolution, it occupies less than a square, and may not appear in the window at all. To change the resolution, use the **Increase Resolution** and **Decrease Resolution** commands in the Configure menu. If you want to force the resolution to the size of the window, use the **Fit To Window** command. You cannot set the resolution to less than 8 bytes per square or more than 1 megabyte per square. The default resolution is 32 bytes per square.

Each square takes up a certain number of pixels in the window. To change the size, use the **Zoom In** and **Zoom Out** commands in the Configure menu. By default, each square is 4 pixels by 4 pixels.

Each type of block is drawn with a different pattern (if the window is in monochrome) or color (if the window is in color). By default, ZoneRanger uses color on color monitors and monochrome on monochrome monitors. To change this setting, choose **Use Color** or **Use Monochrome** from the Configure menu.

In the default color display, free blocks are green, handles are yellow, and pointers are red. If the display is monochrome, free blocks are white, handles are gray, and pointers are black. To change the colors for the blocks, use the Legend Information popup menu. You cannot change the patterns used in a monochrome display.

**See also** "Changing the Graphical windows' colors" on page 34.

**TIP:** Because of the Graphical windows' visual display, you can easily see which blocks are free or relocatable, and which are not. This is great for discovering heap fragmentation.

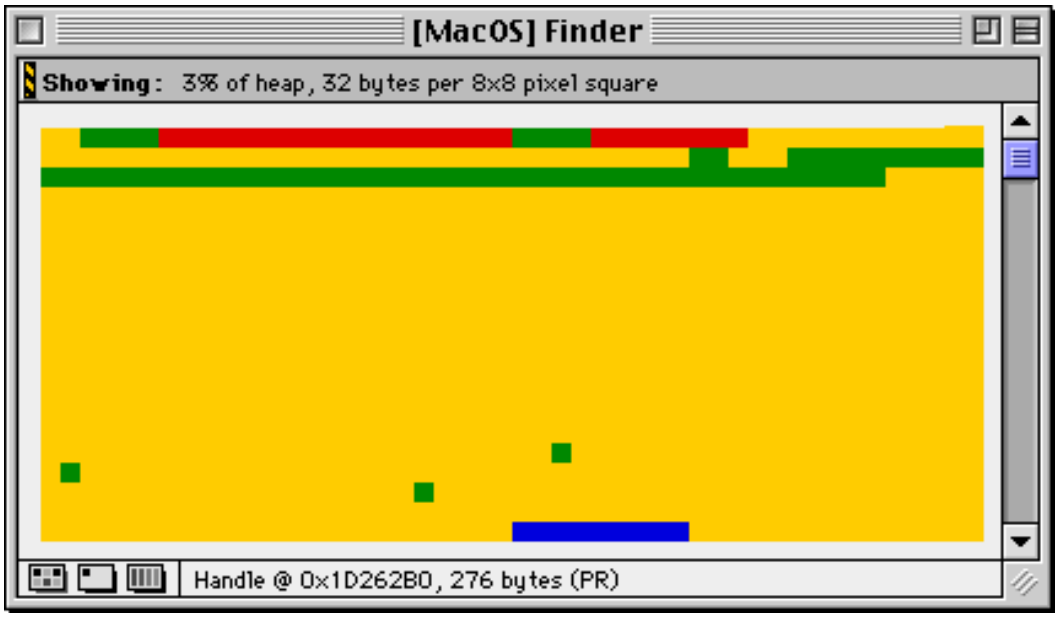
In the ideal application heap, all the locked and non-relocatable blocks are tucked away at either end of the heap. There should be one large free block sitting right in the middle of the heap.

Every time you create a non-relocatable block or lock a relocatable block, you put up a barrier in the heap. If this barrier is in the middle of the heap, it causes problems for the Memory Manager. You may have two mid-sized blocks of free memory divided by the immovable block. If there is a memory request bigger than either of your free blocks, the Memory Manager can't meet the request. It cannot put the two free blocks together because the immovable block sits in between. This is heap fragmentation.

By default, ZoneRanger outlines each block in gray. Outlining helps you get an overall view of how many blocks are in each heap zone and how large they are. To change whether ZoneRanger outlines

each block, use the **Show/Hide Block Outlines** commands in the Configure menu.

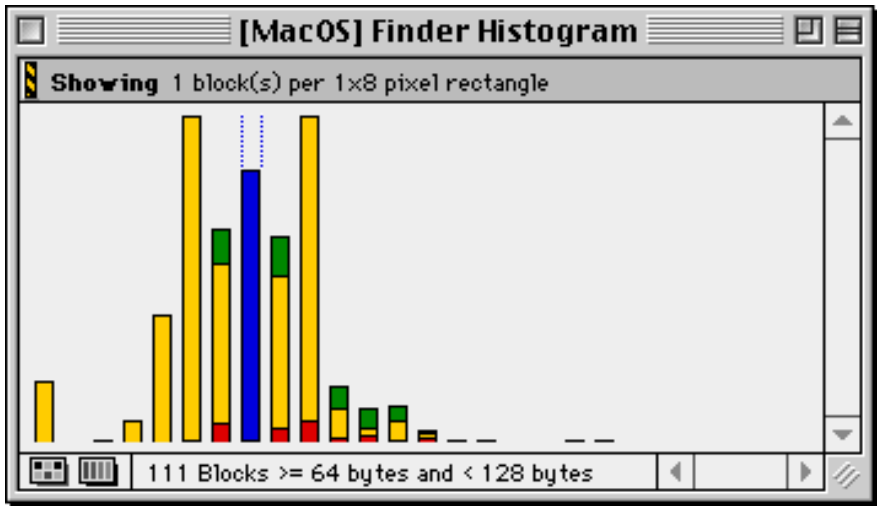
**Figure 3.7 2D Graphical Display with no outlines**




## Histogram Display

This Histogram Display shows a process's heap as a histogram which looks similar to a column chart (Figure 3.8). Many of the commands that work with the 1D and 2D Graphical Displays will work with the Histogram Display. One command that does not work with the Histogram Display is the **Show/Hide Block Outlines** command in the Configure menu.

**Figure 3.8 Histogram window**




The Histogram Display provides a graphical representation of the number of blocks in the heap zone whose size is in each of the various byte ranges shown in Table 3.4. Each column in the histogram provides a visual indication of what percentage of the blocks in that size range are pointers, what percentage are handles and what percentage are free blocks. Click once on a column to display block information in the status bar at the bottom of the window. Double-click a column in the histogram (or select a column and choose **Open Selection** from the File menu) in order to open a Block List Window providing a detailed listing of all blocks in that size range.

**Table 3.4 Byte ranges for Histogram Display**

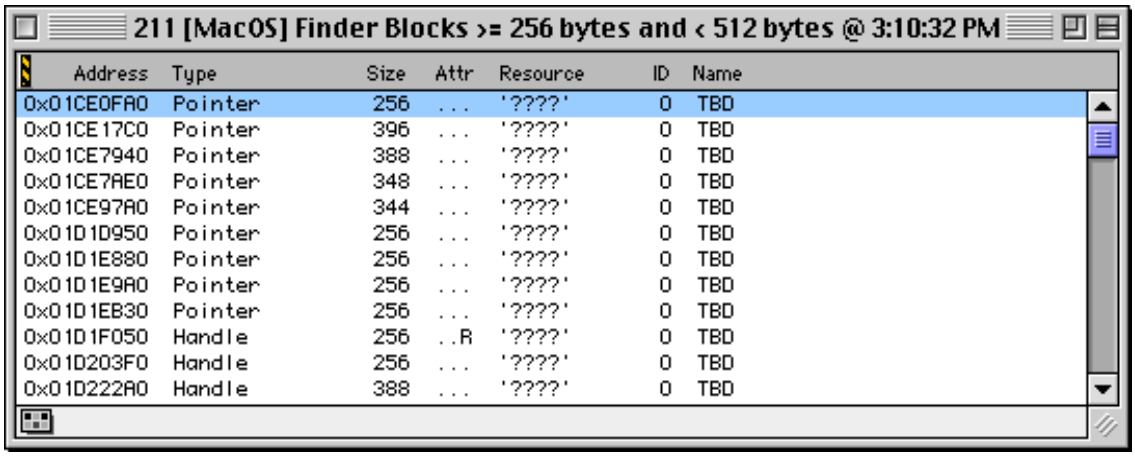
| | | |
|---|---|---|
| 0 | <= size < | 1 |
| 1 | <= size < | 2 |
| 2 | <= size < | 4 |
| 4 | <= size < | 8 |
| 8 | <= size < | 16 |

| | | |
|---|---|---|
| 16 | <= size < | 32 |
| 32 | <= size < | 64 |
| 64 | <= size < | 128 |
| 128 | <= size < | 256 |
| 256 | <= size < | 512 |
| 512 | <= size < | 1K |
| 1K | <= size < | 2K |
| 2K | <= size < | 4K |
| 4K | <= size < | 8K |
| 8K | <= size < | 16K |
| 16K | <= size < | 32K |
| 32K | <= size < | 64K |
| 64K | <= size < | 128K |
| 128K | <= size < | 256K |
| 256K | <= size < | 512K |
| 512K | <= size < | 1024K |
| 1024K | <= size | |

# Block List Window

A Block List Window contains a list of related blocks from an application's heap. ZoneRanger creates a Block List Window when you double-click a column in a Summary Display, double-click a column in the Histogram Display, or find a group of blocks with the **Find** command.

**Figure 3.9 Block List window**


211 [MacOS] Finder Blocks >= 256 bytes and < 512 bytes @ 3:10:32 PM
Address Type Size Attr Resource ID Name
0x01CE0FA0 Pointer 256 ... '????' 0 TBD
0x01CE17C0 Pointer 396 ... '????' 0 TBD
0x01CE7940 Pointer 388 ... '????' 0 TBD
0x01CE7AE0 Pointer 348 ... '????' 0 TBD
0x01CE97A0 Pointer 344 ... '????' 0 TBD
0x01D1D950 Pointer 256 ... '????' 0 TBD
0x01D1E880 Pointer 256 ... '????' 0 TBD
0x01D1E9A0 Pointer 256 ... '????' 0 TBD
0x01D1EB30 Pointer 256 ... '????' 0 TBD
0x01D1F050 Handle 256 ..R '????' 0 TBD
0x01D203F0 Handle 256 ... '????' 0 TBD
0x01D222A0 Handle 388 ... '????' 0 TBD

The name of the window is "*total type description @ time*" where *total* is the number of blocks in this window, *type* is the type of heap, *description* is a brief description of the window's blocks, and *time* is the time you generated the list.

The name of a Block List Window contains the time to remind you that its contents are not updated, unlike the rest of ZoneRanger's windows. Blocks of memory are transient. All types of blocks can be allocated or deallocated at any time. The list of blocks may be outdated mere seconds after creation. There is a Zone Information popup menu in the lower left corner.

The window contains one row for each block. Each row contains seven fields, described in Table 3.5.

**Table 3.5 Block List fields**

| Field | Description |
|---|---|
| Address | The address of the block. |
| Type | The block's type. It can be Pointer, Handle, or Free. |
| Size | The size of the block, in bytes. |

| Field | Description |
|---|---|
| Attr | If the block is a handle, the block's attributes. It can contain L for locked, P for purgeable, and R for resource. |
| Resource | If the block is a resource handle, the type of the resource. |
| ID | If the block is a resource handle, the ID number of the resource. |
| Name | If the block is a resource handle, the name of the resource, if any. |

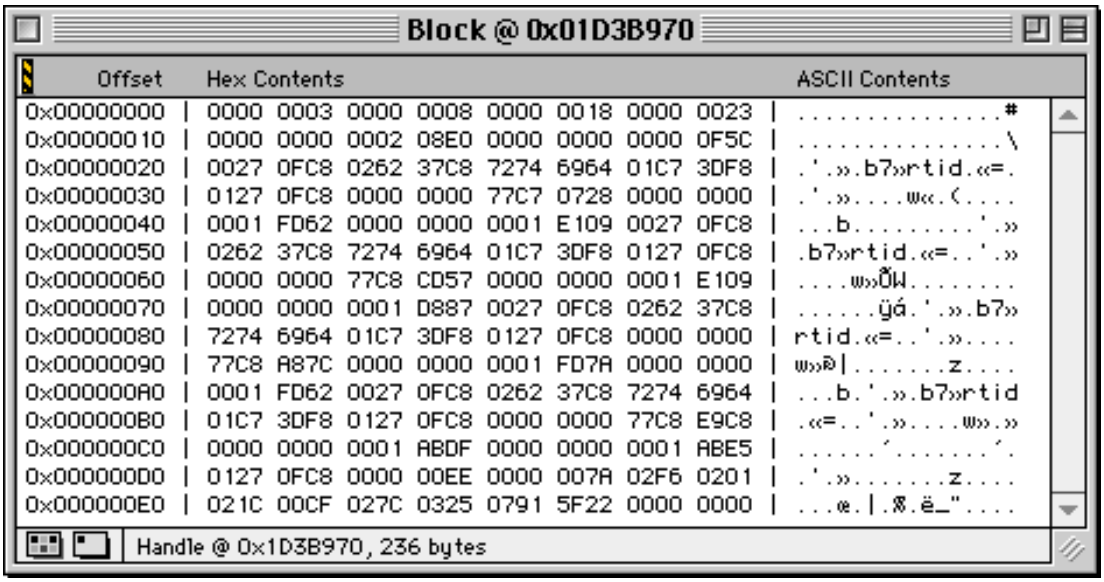
You can copy information on the selected blocks to the clipboard with the **Copy** command or save the information to a file with the **Save As Text** command.

You can also view the contents of the selected blocks in a Block List window. Either double-click on the selection or choose **Open Block @ *address*** from the File menu. ZoneRanger opens a Block Window for each selected block. If the block is no longer valid, the Block Window tells you that the block has been moved or reallocated.

# Block Window

A Block window displays the contents of a block. ZoneRanger opens a Block Window when you double-click on a block in a 1D Graphical Display, 2D Graphical Display, or a Block List Window.

**Figure 3.10 Block Window**




The name of the window is "Block @ *address*" where *address* is the address of the displayed block.

Like the Graphical Windows, the Block window has the Zone and Block Information popup menus in the lower left corner. Also, information about the block is displayed at the bottom of the window.

The Block Window contains the entire contents of the block, 16 bytes per row. Each row has three fields, described in Table 3.6.

**Table 3.6 Block window fields**

| Field | Description |
|---|---|
| Offset | The offset from the beginning of the block to the bytes displayed on this line. |

| Field | Description |
|---|---|
| Hex Contents | The contents of the block, as hexadecimal numbers. |
| ASCII Contents | The contents of the block, as ASCII characters. |

You can copy or save the displayed portion of the block by choosing **Copy** from the Edit menu or Save As Text from the File menu. You cannot copy or save the entire contents of the block, unless the block is small enough to fit entirely in the window.

If the window's block moves or is merged with another block, the window displays the message "Block has moved or been reallocated."

# 4

# Using ZoneRanger

ZoneRanger is a dynamic memory inspection tool that allows you to study how any process—especially one you are creating—uses memory.

## Using ZoneRanger Overview

This chapter discusses the following topics:

- Viewing a Process's Heap—getting the big picture of how a process uses its heap
- Heap Navigation—using the keyboard to navigate the heap
- Searching for Blocks—how to find the blocks you're looking for
- Special Operations—extra features of ZoneRanger

Along the way you'll encounter several tips on how you can use ZoneRanger to accomplish useful tasks. With ZoneRanger you can see how to go about:

- Finding Memory Leaks
- deciding if you are fragmenting memory
- seeing if you're using memory efficiently
- finding memory-related bugs
- Quitting and Relaunching an Application without having to reopen the heap in ZoneRanger.

**NOTE:** You can use ZoneRanger alone, or in combination with the other developer tools available for debugging memory usage. Among the available tools are QC from Onyx Technology, the De-

bugNew library for C++ code (part of the CodeWarrior package), and the "leaks" dcmd for MacsBug. This list is not comprehensive.

# Viewing a Process's Heap

You can view a process's heap in two ways: in a Summary Display or in one or more Graphical Windows.

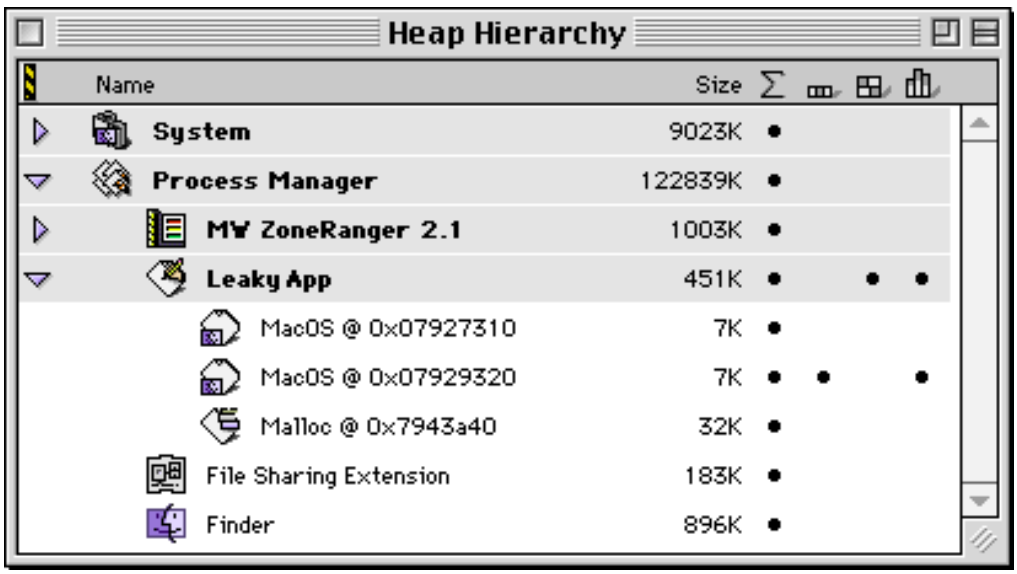
This section discusses:

- Opening a Graphical or Summary Display
- Viewing Block information in Graphical windows
- Viewing Information on a Heap Zone and Block

## Opening a Graphical or Summary Display

Upon launch, ZoneRanger automatically displays the Heap Hierarchy window that lists all the available processes.

**Figure 4.1 The Heap Hierarchy window**



Choose the process whose heap you want to display from the scrolling list. You may select more than one process using standard shift-click or command-click operations. Choose whether to open a 1D Graphical Display, 2D Graphical Display, Summary Display, or Histogram Display—or all four—by activating the bullet markers to the right of the list. To view the heap, double-click the selected process.

As long as a display window remains open, you never have to open that heap again, even if you quit the process and relaunch it. ZoneRanger reconnects the heap automatically when you relaunch the process.

ZoneRanger also remembers each heap's bullet settings. When you relaunch a process, ZoneRanger will restore that process's bullets and hierarchy view to the same way you left them.

**See also** "Choosing How Often to Refresh Contents" on page 43 and "Quitting and Relaunching an Application" on page 50.

## Viewing Block information in Graphical windows

To get more information on a block, click on it and look at the Zone Information popup menu or the Block Information popup menu located in the bottom left corner of the 1D Graphical Display and 2D Graphical Display.

To view the contents of a block, double-click it. ZoneRanger opens a Block Window, described in "Block Window" on page 28.

To save the contents of a Graphical window to a text file or a PICT file, choose **Save As Text** or **Save As Picture** from the File menu.

**See also** "Viewing Information on a Heap Zone and Block" on page 36.

### Displaying different types of handles

In the Graphical Windows, you can highlight one of the three types of relocatable blocks with these commands from the Configure

menu: **Show Locked Handles** , **Show Purgeable Handles** , and **Show Resource Handles** .

An individual block may be of more than one type—for example, a locked resource. You can highlight one or more type of handle at a time. Use the Legend Information popup menu to customize the color for each combination. See "Changing the Graphical windows' colors" on page 34 for more information.

The default colors and patterns for the different types of handles are listed in Table 4.1.

**Table 4.1 Default Handle Colors and Patterns**

| Handle Type | Color | Pattern |
|---|---|---|
| locked | pink | dark gray |
| purgeable | light green | light gray |
| resource | magenta | diamond |

**TIP:** Displaying different kinds of handles allows you to see if you are using memory efficiently. For example, some resources can be very large—PICT resources, sound resources, and so forth. Ideally, these resources should be purgeable so that the Memory Manager can remove them when they are not in use.

By displaying different kinds of handles you can easily see if you have large resource handles that are not purgeable, and discover if you are using memory efficiently.

### Changing the Graphical windows' colors

To change the colors used in the Graphical Windows, click the Legend Information menu in the bottom left corner of the window. You will see the current Legend, illustrated in Figure 4.2.

The Legend Information menu contains seven items that correspond to the colors or patterns used to distinguish the various types of blocks that can be displayed in a heap zone.

If you are using a color display, choose an item to change its color. ZoneRanger displays the Color Picker. You can then choose a different color for that type of block. That color is then used in all Graphical Windows. To restore the colors to their original values, choose **Reset All Colors** .

**TIP:** You can set the color display of the Graphical Windows independently. For example, you could have the 1D Graphical Display use monochrome and the 2D Graphical Display use color.

**Figure 4.2 The Legend Information popup menu**

**NOTE:** If you're displaying a heap in monochrome, you cannot change the patterns used in a Graphical window.

## Viewing Information on a Heap Zone and Block

To get more information on the heap zone in the frontmost window and the selected block, click on the Zone Information or Block Information popup menus. Both of these menus have a "basic" and an "advanced" mode.

### Zone Information popup menu

The Zone Information popup menu contains information about the current heap zone. This menu can be found on the following windows: 1D and 2D Graphical windows, Find dialog, Block List window, and Block window.

**Figure 4.3 Zone Information popup menu**

The meaning of each item is explained in Table 4.2.

**Table 4.2 Heap information**

| Field | Description |
|---|---|
| Heap | The name of the process that contains the heap zone. |
| Address | The address of the heap zone. |
| Size | The size of the heap zone, in kilobytes. |
| Type | The type of memory manager being used for the heap (24-bit, 32-bit, or Modern). |

### Block Information popup menu

The Block Information popup menu contains information about the selected block. If there is no block selected, the menu will say **No valid block** . This menu appears on the Graphical windows and Block window.

**Figure 4.4 Block Information popup menu**

The meaning of each item is explained in Table 4.3.

**Table 4.3 Block information**

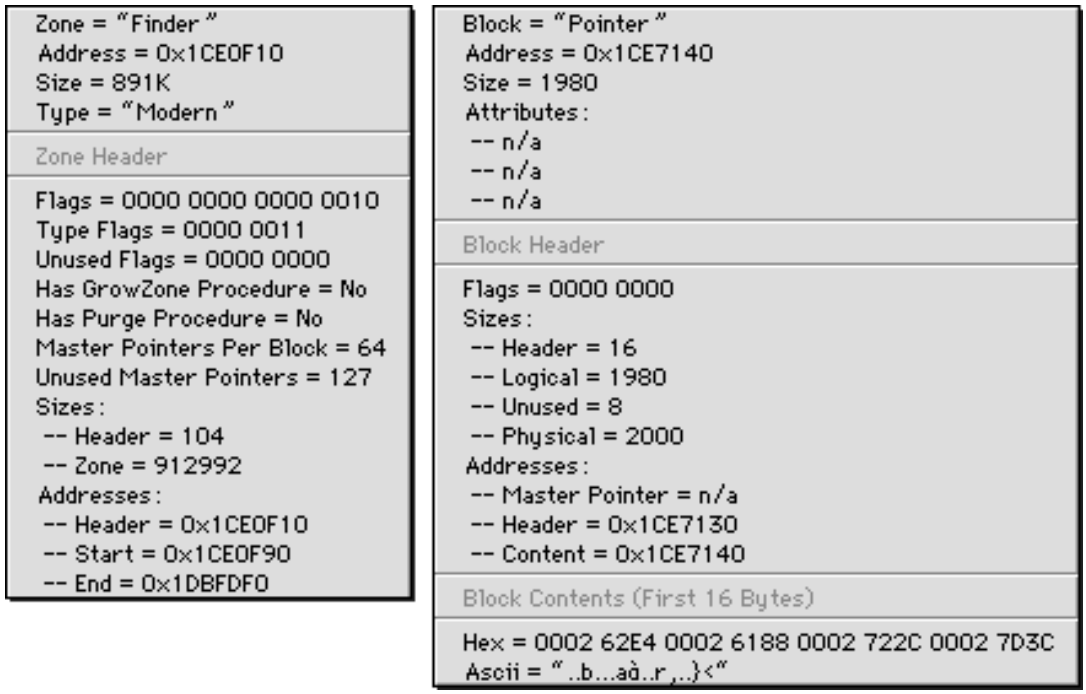
| Field | Description |
|---|---|
| Block | The block's type. |
| Address | The address of the block. |
| Size | The size of the block, in bytes. |
| Attributes | If the block is a handle, the block's attributes (Locked, Purgeable, and Resource). If the block is a resource handle, the type, name, and ID number of the resource is displayed. |

### Advanced Information

If you hold down the option key while clicking on either menu, you will see some "advanced" information about the current zone or selected block, illustrated in Figure 4.5.

**NOTE:** Advanced information is included in ZoneRanger for completeness only. The average user will not have need of this information.

**Figure 4.5 Advanced information**




# Heap Navigation

When you click on a block in the graphical view, that block is selected and information about the block appears at the bottom of the window. You can then use the arrow keys to select a new block and navigate the heap. The functions of the various keys are analogous to those used in the text editor of the CodeWarrior IDE.

**Table 4.4 Keys to Navigate the Heap**

| Key Combination | Description |
|---|---|
| Up Arrow | Move up one row |
| Option-Up Arrow | Move up one page |
| Command-Up Arrow | Move to first block in heap |
| Down Arrow | Move down one row |
| Option-Down Arrow | Move down one page |
| Command-Down Arrow | Move to last block in heap |
| Left Arrow | Move left one block |
| Option-Left Arrow | Move left one block |
| Command-Left Arrow | Move to left end of row |
| Right Arrow | Move right one block |
| Option-Right Arrow | Move right one block |
| Command-Right Arrow | Move to right end of row |

You can use the Page-Up, Page-Down, Home and End keys to scroll the graphical heap view appropriately. Using these keys does not change the current selection.

There are some small differences when using keys to navigate the heap in the 1D Graphical window. These differences are shown in Table 4.5.

**Table 4.5 1D Graphical window key navigation**

| Key Combination | Description |
|---|---|
| Command-Up Arrow | Move to first block in heap |
| Command-Down Arrow | Move to last block in heap |
| Left Arrow | Move left one block |

| Key Combination | Description |
|---|---|
| Option-Left Arrow | Move left one “page” |
| Command-Left Arrow | Move to first block in heap |
| Right Arrow | Move right one block |
| Option-Right Arrow | Move right one “page” |
| Command-Right Arrow | Move to last block in heap |

The Histogram Display supports keyboard navigation by using the arrow keys, shown in Table 4.6.

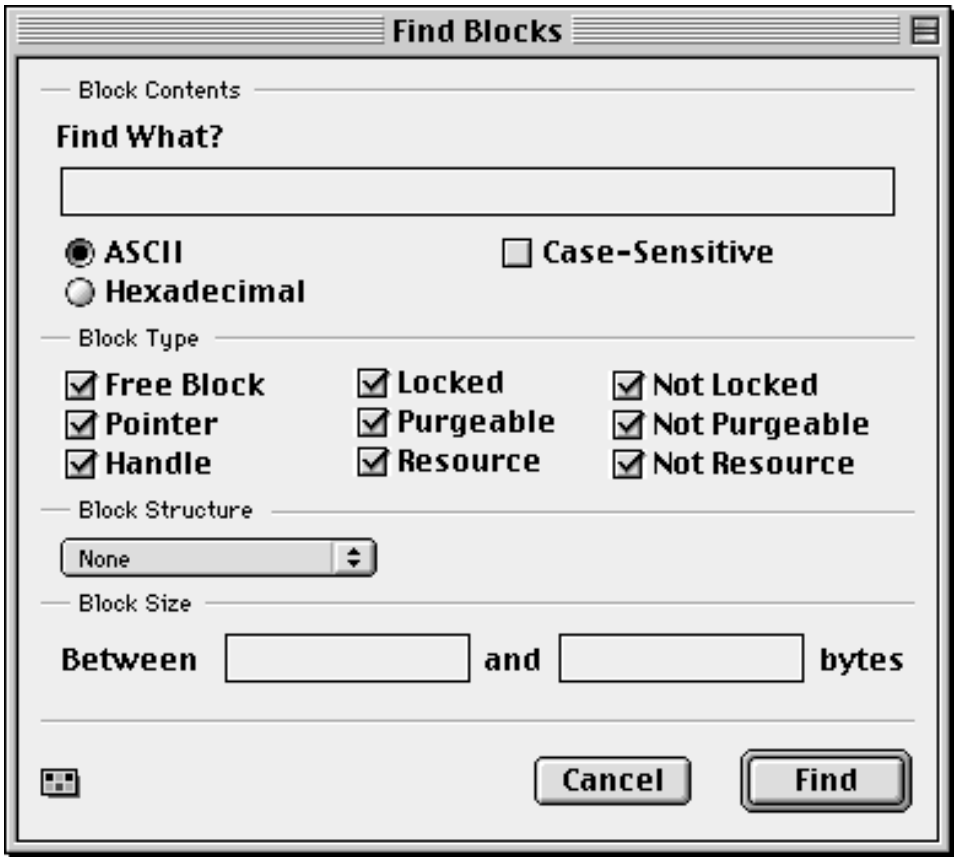
**Table 4.6 Histogram Display navigation keys**

| Key Combination | Description |
|---|---|
| Left Arrow | Move left one column |
| Command-Left Arrow | Move to first column in histogram |
| Right Arrow | Move right one column |
| Command-Right Arrow | Move to last column in Histogram |
| Command-Up Arrow | Move to first column in histogram |
| Command-Down Arrow | Move to last column in histogram |

# Searching for Blocks

ZoneRanger can find blocks based on their contents, attributes, structure, or size. ZoneRanger displays the found blocks in a Block List Window. When you choose **Find** from the Edit menu, Zone–Ranger displays the dialog illustrated in Figure 4.6.

**Figure 4.6 The Find dialog**




The Find dialog has four sections, one for each criterion: Block Contents, Block Type, Block Structure, and Block Size. You can set criteria in one, two, or all four sections. Notice the Zone Information popup menu in the lower left corner.

To search for something in the contents of a block, enter the pattern to search for in the Find What? field. Choose ASCII if you've entered the pattern as an ASCII string. Choose Hexadecimal if you entered the pattern as a hexadecimal number. If you entered an ASCII string, turn on the Case-Sensitive option to make sure that the case of the found string exactly matches the case of the string you entered. Turn off the Case-Sensitive checkbox if the case of string

doesn't matter. If you entered a Hexadecimal string, the Case-Sensitive option has no effect.

Specify the type of blocks to find in the Block Type section. There are nine checkboxes. With the first three options, you can choose the type of blocks to find: free blocks, pointers, or handles. If you turn off all three block types then ZoneRanger finds all types of blocks.

If you are looking for handles, you can also choose what type of handles to find: locked handles, purgeable handles, resource handles, handles that are not locked, not purgeable, or not resources. You must have one or both options set for each block type. For example, assume you want to find all locked handles. You must turn off the "Not Locked" option and leave all of the other five handle options on. If you want to find Locked resource handles, turn off "Not Locked" and "Not Resource" and leave the other options on. Then you find all locked resource handles that are either purgeable or not purgeable. If you leave all six options on, you find all handles.

Choose the structure of the block to find. You can choose one of the blocks illustrated in Figure 4.7.

**Figure 4.7 Block Structure popup menu**

**NOTE:** Blocks in a heap are not marked in any way that would unambiguously indicate the type of structure that is contained in the block. ZoneRanger must therefore make a heuristic determination of the structure based on the size and contents of the block. Since this determination is heuristic and not absolute, it is reasonable to expect that ZoneRanger might miss some true matches and include some false matches.

Specify the range of sizes to find in the Block Size section. Enter the minimum and maximum sizes in bytes. If both fields are empty, then ZoneRanger finds blocks of any size.

---

**TIP:** You can use ZoneRanger's Find command to determine if you are using memory efficiently. Search for large, non-purgeable resource handles, for example. Or, when you know your application is in a state where there should not be blocks of a particular type—locked relocatable blocks for example)—search for them to see if you might have forgotten any.

---

# Special Operations

This section describes how to perform several special tasks in ZoneRanger:

- Choosing How Often to Refresh Contents
- Compacting and Purging Heaps
- Going To and Returning From a Debugger
- Finding Memory Leaks
- Finding Memory Requirements
- Browsing malloc Heaps
- Quitting and Relaunching an Application

## Choosing How Often to Refresh Contents

ZoneRanger lets you choose how often it refreshes the contents of its windows, including the list of running processes in the Heap Hierarchy window. For the heap windows, "refresh the contents" means to reanalyze the heap and display the new results in the window. It does not mean simply drawing the window—the result of a normal window updating process that happens automatically.

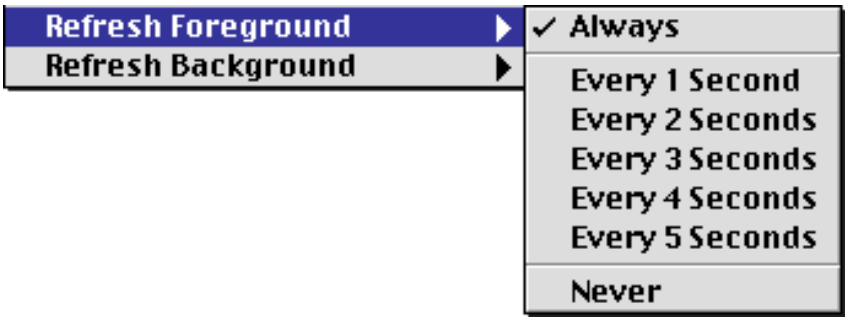
Reanalyzing the heap takes time, so you may not want to do it constantly, especially if ZoneRanger is running in the background. You can set different intervals for refreshing the contents depending on

whether ZoneRanger is the foreground or a background application. Simply choose an interval from the **Refresh Foreground** or **Refresh Background** hierarchical menus, as shown in Figure 4.8. Both menus contain any interval from **1 to 5 seconds**, **Always**, and **Never**.

If you choose **Always**, ZoneRanger refreshes the contents of its windows as often as possible.

If you choose **Never**, ZoneRanger still refreshes the contents of its windows. However, it only does so when actually drawing the particular window—for example, if you move a foreground window and expose a ZoneRanger window.

**Figure 4.8 The Refresh Foreground hierarchical menu**




## Compacting and Purging Heaps

ZoneRanger lets you compact or purge the heap in the frontmost window. These commands are in the Special menu and are available only when a heap is in the frontmost window.

To purge the heap, choose **Purge**. To compact the heap, choose **Compact**. To purge and then compact the heap, choose **Compact and Purge**. When you issue any of these commands, ZoneRanger uses the Mac OS Memory Manager to compact and/or purge the heap.

When the Memory Manager purges a heap, it changes unlocked, purgeable handles to free blocks, but does not move any blocks.

When the Memory Manager compacts a heap, it moves unlocked handles toward the bottom of the heap until they run into a locked or nonrelocatable block, and merges free blocks together.

---

**TIP:** These commands can be very useful for discovering fragmentation problems and for stress-testing your application.
When you compact the heap, all the free and relocatable memory should occupy a contiguous area in the middle of your heap. If, after compacting, there are locked or non-relocatable blocks in the middle of your heap, you've got a fragmentation problem.
If you use purgeable resources without ensuring that they are in memory, you have a very subtle bug that can be hard to locate. By forcibly purging your heap you can discover the problem quickly.

---

## Going To and Returning From a Debugger

To go to a low-level debugger, choose **Debugger** from the Special menu. You break into the installed low-level debugger. If there is no debugger installed or running, the **Debugger** command is disabled—you don't have to worry about crashing.

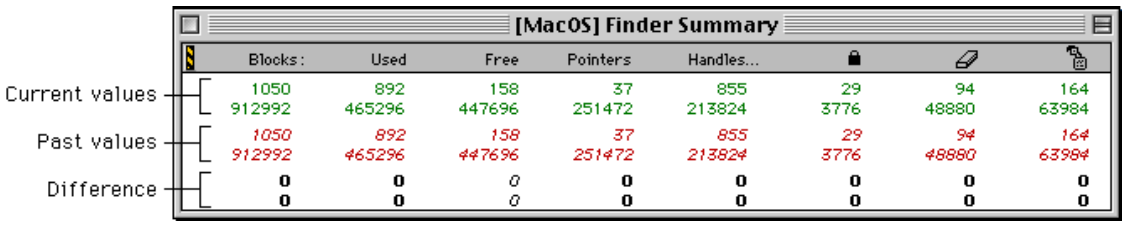
To return to ZoneRanger, use the appropriate `go` command for the debugger. If you are using MacsBug, type `g` and press Return. If you are using The Debugger from Jasik Designs, select **Exit To Program** from the Go menu.

## Finding Memory Leaks

Because the Summary Display gives you the total number of each kind of block, it's great for finding memory leaks.

Launch your application, then open a Summary or Graphical window in ZoneRanger. With either window active, choose **Watch For Leaks** from the ZoneRanger Special menu. When you do, the Summary Display is brought to the front, if it was not active, and expands to display additional data as shown in Figure 4.9.

**Figure 4.9 Watching for leaks in the Summary window**




The values, at the moment you choose **Watch For Leaks** , appear in italic. These don't change—they are the values at the start of the watch. The current numbers appear at the top of the window, in their usual location. These are dynamic and change more or less constantly. The difference between the two appears in bold in the bottom two rows of the Summary Display. This is how the number of blocks has changed since you started the watch.

Look at the number of blocks your process has. Then do things with your application as a normal user would. For example, open and close a window. The point here is to do something, and then return to your initial state. For the **Watch For Leaks** command to be most useful, you should activate it when your application is idling at a well-defined location—typically the main event loop. Then you should exercise the portion of the code that you want to test. Use that code several times in succession.

When you switch back to ZoneRanger, look at the last two rows to determine which type of block—if any— is being leaked.

Ideally, you should have the same number of pointers and handles that you had initially. As a result, the difference data should be zero. If the number of these blocks increases, you may have found a memory leak. Pay particular attention to the numbers of pointers and handles. The number of free blocks (and as a result the total number of blocks) can change at any time. A change in the number of free blocks does not mean you have a leak.

If you have a leak, you can open the appropriate list of blocks. Just double-click on a number in the differences row to open the block list. Typically all the blocks in the leak list are the same size, and the

number of blocks is directly related to the number of iterations that were taken through the leaky code path.

Note, however, that most applications build and destroy blocks at a prodigious rate. Sometimes, just switching from your application to ZoneRanger can cause the numbers to change. What you really want to look for is not whether the difference is zero (although its nice when that happens). What you want to look for is a steady *increase* in the number of pointers or handles after you perform some task and return to your initial state. If that happens, you probably have a memory leak.

### Limitations

ZoneRanger tracks calls that use the Mac OS Memory Manager—calls that use `NewPtr()` or `NewHandle()` directly or indirectly. If you use any other method to allocate memory, ZoneRanger will have a hard time finding leaks.

For example, you may use `NewHandle()` or `NewPtr()` to allocate a large block of memory, and then divide that block into sub-allocations using your own routines. This makes it tougher for ZoneRanger to find leaks. All ZoneRanger knows about is the large allocation using `NewPtr()` or `NewHandle()`. There is no way ZoneRanger can know what memory looks like within that handle or pointer.

As another example, the CodeWarrior default implementation of `operator new` allocates 64K blocks of memory at runtime. Object allocations are created out of those blocks without using the Memory Manager. As a result, the number of blocks do not increase or decrease predictably as you create or dispose of objects.

In addition, if you allocate an object with `new` and forget to `delete` it, you won't be able to find the leak with ZoneRanger because there won't be a spare block left hanging around. The leaking memory is hidden inside the 64K block. You can turn off this default implementation for `new` and `delete` at compile time by setting the `NEW_MODE` compiler flag. Potential values are defined in the "`new.cp`" file, and are listed here for reference.

**Listing 4.1 Memory allocation modes for `operator new`**

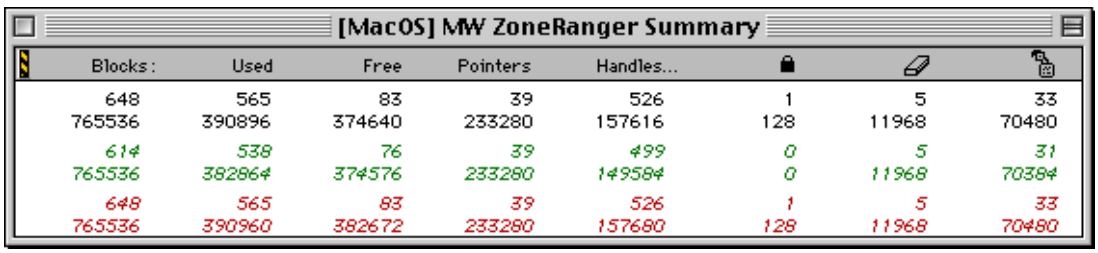
```
#ifndef NEWMODE
#define NEWMODE NEWMODE_FAST // default mode
#endif

#define NEWMODE_NONE   0 // do not define new/delete
#define NEWMODE_SIMPLE 1 // call NewPtr/DisposPtr
#define NEWMODE_MALLOC 2 // use malloc/free
#define NEWMODE_NORMAL 3 // regular new/delete
#define NEWMODE_FAST   4 // regular new/delete
                         // fast version
```

## Finding Memory Requirements

The **Watch For T ides** command is used to help determine the maximum amount of total memory that an application needs to run. You do this by exercising the application and recording the high memory, called high tide, in the Used column of the Summary Display.

Launch your application, then open a Summary or Graphical window in ZoneRanger. With either window active, choose **Watch For Tides** from the Special menu. When you do, the Summary Display of the active heap zone is brought to the front, if it was not active, and expands to display additional data as shown in Figure 4.9.

**Figure 4.10 Watching for high and low tide**




The description of Each row is shown in Table 4.7.

**Table 4.7 Row descriptions**

| Row | Meaning |
|---|---|
| 1 | total amount of each type of block, currently |
| 2 | total size of each type of block, currently |
| 3 | lowest count of each type of block, since start of watch |
| 4 | lowest total size of each type of block, since start of watch |
| 5 | highest count of each type of block, since start of watch |
| 6 | highest total size of each type of block, since start of watch |

Look at the number of blocks your process has. Then do things with your application as a normal user would. For example, open and close a window or file. The point here is to do something, and then return to your initial state. For the **Watch For T ides** command to be most useful, you should activate it when your application is idling at a well-defined location—typically the main event loop. Then you should exercise the portion of the code for which you are interested in the lowest and highest memory usage.

The data shown in Figure 4.10 indicates (among other things):

- the lowest number of pointers at any time was 38
- the lowest total pointer size at any time was 175120
- the highest number of pointers at any time was 39
- the highest total pointer size at any time was 175296

Once **Watch For T ides** has been selected, you can copy the data to the clipboard or save it to a text file.

**See also** "Watch For Tides" on page 59.

## Browsing `malloc` Heaps

A `malloc` heap is not really a "true" heap, but rather a memory pool technique. The MSL and runtime libraries utilize this technique to speed up allocations and minimize "true" heap fragmentation. Browsing `malloc` heaps lets you to view the contents of these "mini-heaps."

See the *MSL C Reference* and the *MSL C++ Reference* for more information about `malloc` heaps.

## Quitting and Relaunching an Application

In a typical scenario, you use ZoneRanger on a project you are developing. You launch the project, then examine the heap in ZoneRanger. In the process, you may discover problems.

To correct the problem, you must quit your project. At that point, ZoneRanger automatically disconnects from your application's heap, because it no longer exists. The ZoneRanger windows will display an appropriate message. The Graphical Windows tell you that the zone has been destroyed. TheSummary Display shows a series of zeroes in the columns. However, the windows remain open unless you close them manually.

If you leave these windows open and subsequently relaunch the application that was the ZoneRanger target, the heap information reappears in the windows. ZoneRanger reconnects to the application automatically. This allows you to find problems, quit the project, correct the problems, and then relaunch the project without having to reopen the heap in ZoneRanger.

# 5

# ZoneRanger Menus

This reference chapter discusses the functionality of each menu item in ZoneRanger.

## ZoneRanger Menus Overview

There are six menus:

- Apple menu—learn about ZoneRanger.
- File menu—opens, closes, and saves windows, and quits the application.
- Edit menu—performs both standard editing and find operations.
- Configure menu—configures the frontmost window or the application.
- Special menu—performs special operations on a heap zone.
- Window menu—lists all open windows.

## Apple menu

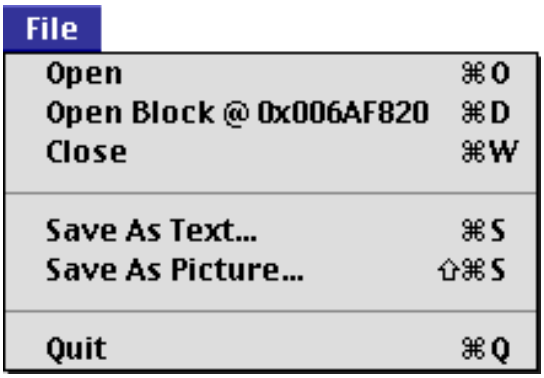
The Apple menu contains one command for ZoneRanger, **About ZoneRanger** . This command displays copyright and author information about the application.

**Figure 5.1 The Apple Menu**

# File menu

The File menu contains commands to open, close, and save windows, and to quit the application.

**Figure 5.2 The File menu**




### Open

Displays the Heap Hierarchy window. In the window, you may open a Graphical and/or a Summary window for selected heap zones.

**See also** "Opening a Graphical or Summary Display" on page 32.

### Open Selection

Opens whatever is currently selected. This item text is dynamic and identifies what you are opening. For example, if you select a block in a Graphical window or Block List window, this item becomes Open Block @ *address*, where *address* is the address of the block. In this example, if you choose this item ZoneRanger opens a Block window for the block.

### Close

Closes the frontmost window.

### Save As Text

Saves the contents of the frontmost window or the currently selected text to a text file.

### Save As Picture

Saves the contents of a Graphical window to a PICT file. The picture contains the entire contents of the heap zone, and not just the currently visible portion. This command is active only if a Graphical window is frontmost.

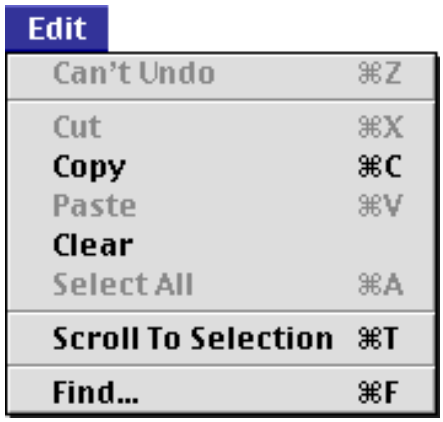
### Quit

Quits the ZoneRanger application.

# Edit menu

The Edit menu contains commands to perform both standard editing and find operations.

**Figure 5.3 The Edit Menu**



### Undo

Is not implemented.

### Cut

This command is not appropriate for ZoneRanger.

### Copy

This command functions differently depending on which window is front-most. If a Block Window is front-most, this command copies the visible area of the window to the clipboard. If any other ZoneRanger window is front-most, this command copies the current selection to the clipboard. The Copy command cannot be used with the Histogram Display.

### Paste

This command is not appropriate for ZoneRanger.

### Clear

Clears (deselects) the current selection in the active Graphical window.

### Select All

Performs the standard editing operation. This command is only available in the Block List Window.

### Scroll To Selection

Scrolls the selected block into view, if it is not already in view. This command is enabled when you select a block from the heap (in graphical view).

### Find

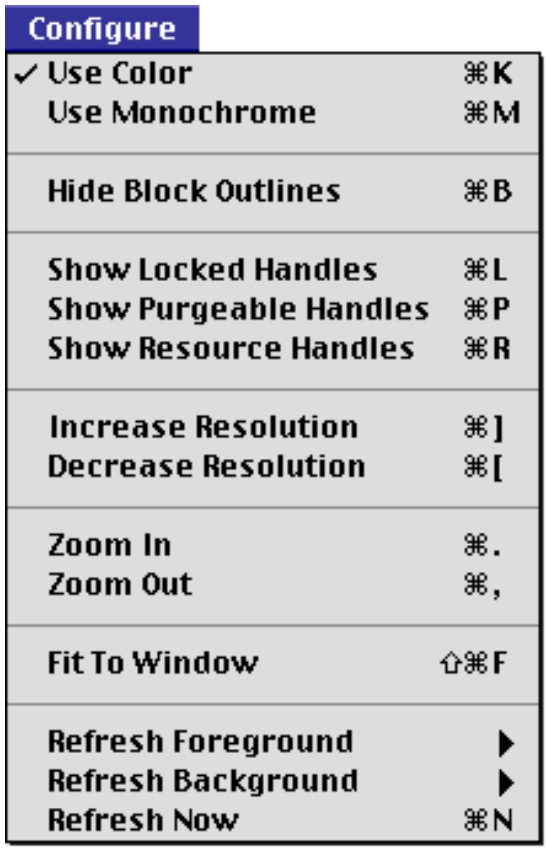
Opens the Find dialog box. This dialog lets you find blocks based on their content, type, or size.

**See also** "Searching for Blocks" on page 40.

# Configure menu

The Configure menu contains commands to configure the *frontmost* Graphical window, and to set how often ZoneRanger refreshes window contents. Except for the Refresh commands, each of the Configure menu items is enabled only if the frontmost window is a ZoneRanger Graphical window.

**Figure 5.4 The Configure menu**




### Use Color

Displays the frontmost Graphical window in color.

### Use Monochrome

Displays the frontmost Graphical window in monochrome.

### Show/Hide Block Outlines

**Show Block Outlines** sets the frontmost Graphical window to display the boundaries of each block in the zone. If block outlines are currently being shown, then this menu item becomes **Hide Block Outlines** .

**See also** "Displaying different types of handles" on page 33.

In a Graphical window, handles are displayed in a particular color or pattern. The next three items—**Show Locked Handles** , **Show Purgeable Handles** , and **Show Resource Handles** —allow you to highlight one particular type of handle in its own color or pattern. A check mark next to the item indicates which type of handle, if any, is highlighted. Only one of these three items may be active at any time.

**See also** "Changing the Graphical windows' colors" on page 34.

### Increase Resolution

Doubles the resolution of the frontmost Graphical window by halving the number of bytes that each square represents. The highest resolution is 8 bytes per square. This command is available only if the resolution is not already at the maximum.

### Decrease Resolution

Halves the resolution of the frontmost Graphical window by doubling the number of bytes that each square represents. The lowest resolution is 1 megabyte per square. This command is available only if the resolution is not already at the minimum.

### Zoom In

Increase the size of each grid square in the display. The allowable zoom factors are 1x1, 4x4, and 8x8 pixel grid squares. The 1x1 pixel zoom factor is not available if you've selected **Show Block Outlines**, because a one pixel square is not large enough to draw the

outline of the block. The **Zoom In** command is available only if the zoom factor is not already at the maximum.

### Zoom Out

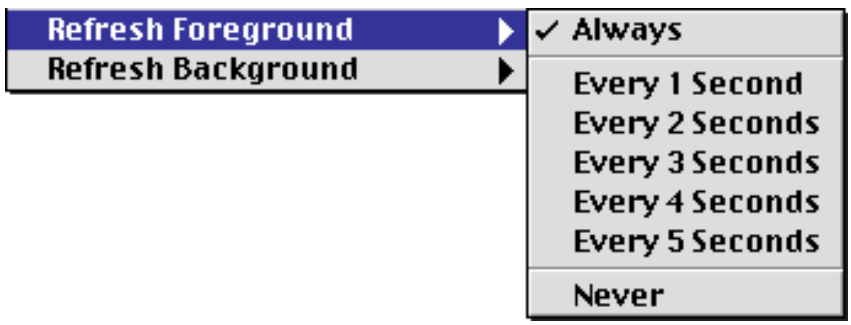
Decrease the size of each grid square in the display. The allowable zoom factors are 1x1, 4x4, and 8x8 pixel grid squares. The 1x1 pixel zoom factor is not available if you've selected **Show Block Outlines**, because a one pixel square is not large enough to draw the outline of the block. The **Zoom Out** command is available only if the zoom factor is not already at the minimum.

### Fit To Window

Changes the resolution in the Graphical window such that the entire heap is displayed within the current size of the window. The Fit To Window command is available only if the resolution is higher than the window can currently display.

The **Refresh Foreground** and **Refresh Background** menu items each present a submenu with which you can specify how often the application will refresh the contents of windows when the application is in the foreground or background, respectively. You can choose a value in the range from 1 to 5 seconds, as well as **Always** or **Never**.

**Figure 5.5 The Refresh Foreground hierarchical menu**



If you choose **Always**, ZoneRanger refreshes the contents of its windows as often as possible.

If you choose **Never**, ZoneRanger still refreshes the contents of its windows. However, it only does so when actually drawing the particular window—for example, if you move a foreground window and expose the ZoneRanger window.

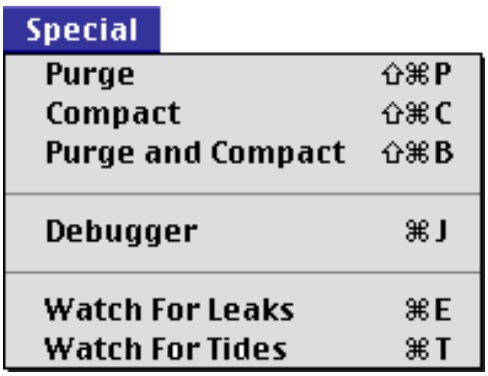
### Refresh Now

Updates a window immediately. Choose this command if you do not want to wait for the refresh interval to elapse. This command is particularly useful if you've set the refresh rate to **Never**.

# Special menu

The Special menu contains commands that perform special operations on a heap zone.

**Figure 5.6 The Special menu**




### Compact

Compacts the heap zone by moving unlocked handles toward the bottom of the heap, and merging free blocks together. This command is available only if a Summary or Graphical window is frontmost.

**See also** "Compacting and Purging Heaps" on page 44.

### Purge

Purges the heap zone by changing unlocked, purgeable handles to free blocks. This command is available only if a Summary or Graphical window is frontmost.

### Compact and Purge

Purges the heap zone and then compacts it. This command is available only if a Summary or Graphical window is frontmost.

### Debugger

Puts you into the installed low-level debugger. If there is no debugger installed or running, the Debugger command is disabled—you don't have to worry about crashing.

To return to ZoneRanger, use the appropriate go command for the debugger. If you are using MacsBug, type `g` and press Return. If you are using The Debugger from Jasik Designs, choose **Exit To Program** from the **Go** menu.

**See also** "Going To and Returning From a Debugger" on page 45.

### Watch For Tides

**Watch For Tides** expands the Summary Display to show your applications high and low memory usage (high tide and low tide). This command becomes **Stop Watching For Tides** if you are already in watch mode. When you choose Watch For Tides with one of the Graphical Windows front-most, ZoneRanger brings the corresponding Summary Display to the front and begins watching for high and low memory usage in that heap zone.

This command and **Watch For Leaks** are mutually exclusive. Only one of these commands can be active at a time.

**See also** "Finding Memory Requirements" on page 48.

**Watch For Leaks**

**Watch For Leaks** expands the Summary Display to show a comparison between blocks at one point in time and blocks at another. If you are already in watch mode, this item becomes **Stop Watching For Leaks** . When you choose **Watch For Leaks** with one of the Graphical Windows frontmost, ZoneRanger brings the corresponding Summary Display to the front, and begins watching for leaks in that heap zone.

This command and **Watch For T ides** are mutually exclusive. Only one of these commands can be active at a time.

**See also** "Finding Memory Leaks" on page 45.

# Window menu

The Window menu contains one menu item for each open window (graphical, summary, block list, and block windows). When you choose a window from this menu, ZoneRanger activates the window. ZoneRanger adds windows to the list in the order in which they were opened. The first nine windows have the command key equivalents 1 through 9 respectively.

**Figure 5.7 The Window menu**

# Index


## A

Advanced Information 37

## B

block attributes 12
Block Information popup menu 36
    Advanced information 37
Block List window 26
block outlines 21, 24
block size 12
block type
    defined 12
    displaying 33
Block window 28
blocks
    finding 40
    information on 36
    listing 26
    viewing contents 28

## C

C++
    memory allocation modes for new 48
Close command (ZoneRanger) 52
CodeWarrior
    described 6
colors
    customizing 34
    restoring 35
Compact and Purge command (ZoneRanger) 44, 59
Compact command (ZoneRanger) 44, 58
customizing
    refresh interval 43

## D

Debugger command (ZoneRanger) 45, 59
Decrease Resolution command (ZoneRanger) 21, 22, 56

## F

Find command 40
finding blocks 40
Fit To Window command (ZoneRanger) 21, 22, 57
fragmented heap 23, 45
free blocks 12

## G

Graphical window
    1D Display 20
    2D Display 22
Graphical windows 18–36

## H

handle
    defined 13
    locked 12, 34
    relocatable block 12
    resource 12
heap
    compacting 44
    defined 11
    fragmentation 23, 45
    information on 36
    purging 44
    viewing 32
Heap Hierarchy window 16, 32
Hide Block Outlines command 21, 24, 56
Hide Block Outlines command (ZoneRanger) 21, 24

## I

Increase Resolution command (ZoneRanger) 21, 22, 56

## J

Jasik's The Debugger
    and ZoneRanger 45

## K

Keys
    End 39
    Home 39
    Page-Down 39
    Page-Up 39
    To navigate the Heap 39

## L

Legend Information popup menu 34
locked handles 12, 34

## M

MacsBug
    and ZoneRanger 45
malloc 50
memory leak 45
Menus
    Block Information 36
    Legend Information 34
    Zone Information 36

## N

NEWMODE compiler flag 47
non-relocatable blocks 12

## O

Open command (ZoneRanger) 52
Open Selection command (ZoneRanger) 52
opening
    Block List window 26
    Block window 28
operator new
    memory allocation modes 48
    with ZoneRanger 47
outlines, block 21, 24

## P

pointer
    defined 12
    non-reloctable block 12
process 10
Purge command (ZoneRanger) 44, 59
purgeable handles 12, 34
purgeable resources 45

## Q

Quit command (ZoneRanger) 53

## R

Refresh Background command (ZoneRanger) 44, 57
Refresh Foreground command (ZoneRanger) 44, 57
refresh interval, customizing 43
Refresh Now command (ZoneRanger) 58
relocatable blocks 12
resolution 21, 22, 56
resource handles 12, 34

## S

Save as Picture command (ZoneRanger) 33, 53
Save as Text command (ZoneRanger) 28, 33, 53
Scroll To Selection command (ZoneRanger) 54
searching for blocks 40
Show Block Outlines command 21, 24, 56
Show Block Outlines command (ZoneRanger) 24
Show Locked Handles command (ZoneRanger) 34, 56
Show Purgeable Handles command (ZoneRanger) 34, 56
Show Resource Handles command (ZoneRanger) 34, 56
Stop Watching For Leaks 60
Stop Watching For Tides 59
Summary window 17

## U

Use Color command (ZoneRanger) 21, 23, 55
Use Monochrome command (ZoneRanger) 21, 23, 56

## W

Watch for Leaks command 60
Watch for Leaks command (ZoneRanger) 45
Watch For Tides 48, 59
what's in this manual 7

## Z

Zone Information popup menu 36
    Advanced information 37
Zoom In command (ZoneRanger) 21, 23, 56

Zoom Out command (ZoneRanger) 21, 23, 57

# CodeWarrior

# ZoneRanger User Guide

## Credits


| | |
|---|---|
| **writing lead:** | Derek Saldana |
| **other writers:** | Carl B. Constantine, Jeff Mattson, Marc Paquette, Roger Wong |
| **engineering:** | Joshua Golub |
| **frontline warriors:** | CodeWarrior users everywhere |

# Guide to CodeWarrior Documentation

CodeWarrior documentation is modular, like the underlying tools. There are manuals for the core tools, languages, libraries, and targets. The exact documentation provided with any CodeWarrior product is tailored to the tools included with the product. Your product will not have every manual listed here. However, you will probably have additional manuals (not listed here) for utilities or other software specific to your product.

| **Core Documentation** | |
|---|---|
| IDE User Guide | How to use the CodeWarrior IDE |
| Debugger User Guide | How to use the CodeWarrior debugger |
| CodeWarrior Core Tutorials | Step-by-step introduction to IDE components |
| **Language/Compiler Documentation** | |
| C Compilers Reference | Information on the C/C++ front-end compiler |
| Pascal Compilers Reference | Information on the Pascal front-end compiler |
| Error Reference | Comprehensive list of compiler/linker error messages, with many fixes |
| Pascal Language Reference | The Metrowerks implementation of ANS Pascal |
| Assembler Guide | Stand-alone assembler syntax |
| Command-Line Tools Reference | Command-line options for Mac OS and Be compilers |
| Plugin API Manual | The CodeWarrior plugin compiler/linker API |
| **Library Documentation** | |
| MSL C Reference | Function reference for the Metrowerks ANSI standard C library |
| MSL C++ Reference | Function reference for the Metrowerks ANSI standard C++ library |
| Pascal Library Reference | Function reference for the Metrowerks ANS Pascal library |
| MFC Reference | Reference for the Microsoft Foundation Classes for Win32 |
| Win32 SDK Reference | Microsoft's Reference for the Win32 API |
| The PowerPlant Book | Introductory guide to the Metrowerks application framework for Mac OS |
| PowerPlant Advanced Topics | Advanced topics in PowerPlant programming for Mac OS |
| **Targeting Manuals** | |
| Targeting BeOS | How to use CodeWarrior to program for BeOS |
| Targeting Java VM | How to use CodeWarrior to program for the Java Virtual Machine |
| Targeting Mac OS | How to use CodeWarrior to program for Mac OS |
| Targeting MIPS | How to use CodeWarrior to program for MIPS embedded processors |
| Targeting NEC V810/830 | How to use CodeWarrior to program for NEC V810/830 processors |
| Targeting Net Yaroze | How to use CodeWarrior to program for Net Yaroze game console |
| Targeting Nucleus | How to use CodeWarrior to program for the Nucleus RTOS |
| Targeting OS-9 | How to use CodeWarrior to program for the OS-9 RTOS |
| Targeting Palm OS | How to use CodeWarrior to program for PalmOS |
| Targeting PlayStation OS | How to use CodeWarrior to program for the PlayStation game console |
| Targeting PowerPC Embedded Systems | How to use CodeWarrior to program for PPC embedded processors |
| Targeting VxWorks | How to use CodeWarrior to program for the VxWorks RTOS |
| Targeting Win32 | How to use CodeWarrior to program for Windows |